

# Stochastic Coordinate Coding and Its Application for *Drosophila* Gene Expression Pattern Annotation

Binbin Lin<sup>1</sup>, Qingyang Li<sup>1</sup>, Qian Sun<sup>1</sup>, Ming-Jun Lai<sup>2</sup>, Ian Davidson<sup>3</sup>, Wei Fan<sup>4</sup>, Jieping Ye<sup>1</sup>

<sup>1</sup>Center for Evolutionary Medicine and Informatics, The Biodesign Institute, ASU, Tempe, AZ

<sup>2</sup>Department of Mathematics, University of Georgia, Athens, GA

<sup>3</sup>Department of Computer Science, University of California, Davis, CA

<sup>4</sup>Noah's Ark Lab, Huawei Technologies Co. Ltd., Sha Tin, Hong Kong

**Abstract**—*Drosophila melanogaster* has been established as a model organism for investigating the fundamental principles of developmental gene interactions. The gene expression patterns of *Drosophila melanogaster* can be documented as digital images, which are annotated with anatomical ontology terms to facilitate pattern discovery and comparison. The automated annotation of gene expression pattern images has received increasing attention due to the recent expansion of the image database. The effectiveness of gene expression pattern annotation relies on the quality of feature representation. Previous studies have demonstrated that sparse coding is effective for extracting features from gene expression images. However, solving sparse coding remains a computationally challenging problem, especially when dealing with large-scale data sets and learning large size dictionaries. In this paper, we propose a novel algorithm to solve the sparse coding problem, called Stochastic Coordinate Coding (SCC). The proposed algorithm alternatively updates the sparse codes via just a few steps of coordinate descent and updates the dictionary via second order stochastic gradient descent. The computational cost is further reduced by focusing on the non-zero components of the sparse codes and the corresponding columns of the dictionary only in the updating procedure. Thus, the proposed algorithm significantly improves the efficiency and the scalability, making sparse coding applicable for large-scale data sets and large dictionary sizes. Our experiments on *Drosophila* gene expression data sets demonstrate the efficiency and the effectiveness of the proposed algorithm.

## I. INTRODUCTION

*Drosophila melanogaster* has been established as a model organism for investigating the fundamental principles of developmental gene interactions [28], [18], [23]. The Berkeley *Drosophila* Genome Project (BDGP, [34], [35]) has produced a comprehensive atlas of gene expression patterns in the form of digital images by RNA *in situ* hybridization [13] in order to facilitate a deep understanding of transcriptional regulation during *Drosophila* embryogenesis. The images in BDGP are annotated with anatomical and developmental ontology terms using a controlled vocabulary [34]. To facilitate pattern discovery and comparison, many web-based resources have been created to conduct comparative analysis based on the body part keywords and the associated images. Currently, the annotation is performed manually by human experts. With the increasing size of available images generated by high throughput technologies, it is imperative to design efficient and effective computational methods to automatically annotate

images capturing spatial patterns of gene expression.

The BDGP gene expression pattern annotation problem can be formulated as an image annotation problem, which has been widely studied in computer vision and machine learning. In particular, a collection of images from the same developmental stage range and the same gene are annotated by a sub-set of the keywords (see Fig. 1). Although traditional image annotation methodologies can be employed to solve this problem, significant challenges remain due to the multi-instance multi-label nature of this problem. Since the annotation associated with a group of images does not imply an association with all the images in this group, we need to develop approaches to retain the group membership information. Due to the effects of stochastic processes during embryogenesis, no two embryos develop identically. And the current image acquisition techniques limit the quality of the images. Thus, the shape and color of the same body part may vary from image to image. Invariance to local distortions is required for an accurate annotation system. Several prior works on the automatic annotation of *Drosophila* gene expression images have been reported. Zhou and Peng [39] constructed their system based on the assumption that each image in the group is annotated by all the terms assigned to that group; Ji et al. [16] considered a learning framework that incorporates the term-term interactions; Yuan et al. [38] and Sun et al. [30] adopted sparse coding for image annotation with the dictionary generated by the clustering techniques. However, the dictionary is fixed during the training process due to the expensive learning cost. This motivates us to develop an efficient sparse coding algorithm to efficiently learn the dictionary and sparse feature representations from the data.

Sparse coding concerns the problem of reconstructing data vectors using sparse linear combinations of basis vectors [24], [8], [11]. It has become extremely popular for learning the dictionary and extracting features from images in the last decade. Sparse coding has been applied in many fields including audio processing [29], text mining [1] and image recognition [31]. Different from traditional feature extraction methods like principal component analysis and its variants, sparse coding learns non-orthogonal and over-complete dictionaries which have more flexibility to represent the data. Sparse coding can also model inhibition between the bases by sparsifying

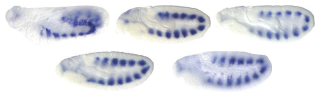
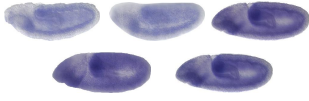
Bag of images	BDGP terms
	Fat body/gonad primordium Somatic muscle primordium
	Anterior endoderm primordium Head mesoderm primordium P2 Inclusive hindgut primordium Posterior endoderm primordium Trunk mesoderm primordium

Fig. 1. Sample bag (groups) of images and the associated terms in BDGP database.

their activations. Similar properties have been observed in biological neurons, thus making sparse coding a plausible model of the visual cortex [25], [26].

Despite the rich promise of sparse coding models, sparse coding is computationally expensive especially when dealing with large-scale data. The main computational cost of sparse coding lies in the updating of sparse codes and the dictionary. It is known that updating the sparse code is usually much more time consuming. Therefore, much of recent work has been devoted to seeking efficient optimization algorithms for updating the sparse code [17], [33]. The basic idea of these methods is to quickly identify the non-zero entries of the sparse code, thus reducing the search space. However, most of these algorithms are iterative batch methods which may not scale to very large data sets [6], since updating the dictionary involves the computation of the full gradient of the dictionary from the whole data set and is expensive. Recently, several work based on stochastic gradient descent and online learning has been proposed. [21] proposed an online dictionary learning algorithm which updates the dictionary for each incoming data point. It is expected that the dictionary will converge faster in the online setting. However, even when the dictionary has been learned, one has to further learn the sparse code, which is also computationally expensive especially for large-scale data sets.

In this paper, we propose a novel approach for efficiently solving the sparse coding problem. The key ingredients of the proposed SCC algorithm involve:

- 1) When updating the sparse code, we only perform a few steps of coordinate descent. For each image patch, we further speed up the coordinate descent by updating only the support of the sparse code.
- 2) When updating the dictionary, we only update the columns corresponding to the support of the sparse code.
- 3) When doing stochastic gradient descent, we choose an adaptive learning rate which speeds up the convergence of the algorithm.

Extensive experiments on Drosophila gene expression data sets demonstrate the efficiency of the proposed algorithm.

## II. BACKGROUNDS AND RELATED WORK

In this section, we review sparse coding and related work.

We first introduce our notation used throughout this paper. We use boldface lower case letters (e.g.,  $\mathbf{x}, \mathbf{z}$ ) to denote vectors and use boldface upper case letters (e.g.,  $\mathbf{X}, \mathbf{Z}$ ) to denote matrices. Scalars are denoted by lower or upper case letters (e.g.,  $p, M$ ). Given a data set  $\mathbf{X} = (\mathbf{x}_1 \cdots \mathbf{x}_n)$  of image patches, each image patch is a  $p$ -dimensional vector, i.e.,  $\mathbf{x}_i \in \mathbb{R}^p$ ,  $i = 1, \dots, n$ . Moreover, each  $\mathbf{x}_i$  is preprocessed to be zero mean and unit  $l_2$  norm. We first extract meaningful features from these image patches using sparse coding. The learned features will be used for image annotation.

The linear decomposition of an image patch using a few number of basis or atoms of a learned dictionary has recently led to state-of-art performance in numerous signal processing and machine learning tasks. Specifically, suppose there are  $m$  atoms  $\mathbf{d}_j \in \mathbb{R}^p$ ,  $j = 1, \dots, m$ , where the number of atoms is usually much smaller than the number of image patches  $n$  but larger than the dimension of the image patch  $p$ . Each image patch can then be represented as  $\mathbf{x}_i = \sum_{j=1}^m z_{i,j} \mathbf{d}_j$ . Therefore, each  $p$ -dimensional image patch  $\mathbf{x}_i$  is represented by a  $m$ -dimensional vector  $\mathbf{z}_i = (z_{i,1}, \dots, z_{i,m})^T$ . It is further assumed that each image patch can be represented only by a small group of atoms, that is, the learned feature vector  $\mathbf{z}_i$  is a sparse vector.

Given one image patch  $\mathbf{x}_i$ , one can formularize the above idea as the following optimization problem:

$$\min f_i(\mathbf{D}, \mathbf{z}_i) = \frac{1}{2} \|\mathbf{D}\mathbf{z}_i - \mathbf{x}_i\|^2 + \lambda \|\mathbf{z}_i\|_1, \quad (1)$$

where  $\lambda$  is the regularization parameter,  $\|\cdot\|$  is the standard Euclidean norm and  $\|\mathbf{z}_i\|_1 = \sum_{j=1}^m |z_{i,j}|$ . The first term of Eq.(1) is the reconstruction error, which measures how well the new feature represents the image patch. The second term of Eq.(1) ensures the sparsity of the learned feature  $\mathbf{z}_i$ . Each  $\mathbf{z}_i$  is often called the *sparse code*. Since  $\mathbf{z}_i$  is sparse, there are only a few entries in  $\mathbf{z}_i$  which are non-zero. We call its non-zero entries as its *support*, i.e.,  $\text{supp}(\mathbf{z}_i) = \{z_{i,j} : z_{i,j} \neq 0, j = 1, \dots, m\}$ . Here  $\mathbf{D} = (\mathbf{d}_1 \cdots \mathbf{d}_m) \in \mathbb{R}^{m \times p}$  is called the *dictionary*. To prevent an arbitrary scaling of the sparse code, each column of  $\mathbf{D}$  is restricted to be in a unit ball, i.e.,  $\|\mathbf{d}_j\| \leq 1$ . Given the whole data set  $\mathbf{X} = (\mathbf{x}_1 \cdots \mathbf{x}_n)$ , the sparse coding problem is then given as follows:

$$\min_{\mathbf{D} \in B_m, \mathbf{z}_1, \dots, \mathbf{z}_m} \mathcal{F}(\mathbf{D}, \mathbf{z}_1, \dots, \mathbf{z}_m) \equiv \frac{1}{n} \sum_{i=1}^n f_i(\mathbf{D}, \mathbf{z}_i), \quad (2)$$

where  $B_m$  is the feasible set of  $\mathbf{D}$  which is defined as follows:

$$B_m = \{\mathbf{D} \in \mathbb{R}^{p \times m} : \forall j = 1, \dots, m, \|\mathbf{d}_j\|_2 \leq 1\}.$$

It is a non-convex problem with respect to joint parameters in the dictionary  $\mathbf{D}$  and the sparse codes  $\mathbf{Z} = (\mathbf{z}_1 \cdots \mathbf{z}_n)$ . Therefore, it is often difficult to find a global optimum. However, it is a convex problem when either  $\mathbf{D}$  or  $\mathbf{Z}$  is fixed. When the dictionary  $\mathbf{D}$  is fixed, solving each sparse code  $\mathbf{z}_i$  is the well known lasso problem [32]. Many methods have been proposed to solve this problem, including Least Angle Regression (LARS, [12]), Fast Iterative Soft-Thresholding Algorithm (FISTA, [3]) and Coordinate Descent (CD, [37]).

It might be worth noting that when the feature dimension  $m$  is large which is often the case, solving a lasso problem is very time consuming. When the sparse codes are fixed, it is a simple quadratic problem. Therefore, one often uses an alternating optimization approach to solve the sparse coding problem. Specifically, when  $\mathbf{D}$  is fixed, we update the sparse code  $\mathbf{z}_i$  for each image patch  $\mathbf{x}_i$ . When the sparse codes are fixed, we use gradient descent to update the dictionary:

$$\mathbf{D} \leftarrow \mathbf{D} - \eta \frac{1}{n} \sum_{i=1}^n \nabla_{\mathbf{D}} f_i(\mathbf{D}, \mathbf{z}_i) = \mathbf{D} - \eta \frac{1}{n} \sum_{i=1}^n (\mathbf{D} \mathbf{z}_i - \mathbf{x}_i) \mathbf{z}_i^T,$$

where  $\eta$  is the step size. However, at each iteration, full gradient descent requires evaluation of  $n$  derivatives, which is very expensive when the data set is of large-scale. A popular modification is Stochastic Gradient Descent (SGD, see [5]). At each iteration, we randomly draw an image patch  $\mathbf{x}_t$ , and update the dictionary as follows:

$$\mathbf{D}_{t+1} \leftarrow \mathbf{D}_t - \eta_t \nabla_{\mathbf{D}_t} f_t(\mathbf{D}_t, \mathbf{z}_t),$$

where  $\eta_t$  is called the learning rate.

We summarize the optimization methods in the following. First we initialize the dictionary  $\mathbf{D}$ . Many dictionary initialization methods have been proposed, such as random weights [14], random patches and k-means. A detailed comparison of the performance among these initialization methods has been discussed in [9]. With the initial dictionary, conventional sparse coding algorithms include the following main steps:

- 1) Get an image patch  $\mathbf{x}_i$ .
- 2) Calculate the sparse code  $\mathbf{z}_i$  by using LARS, FISTA or coordinate descent.
- 3) Update the dictionary  $\mathbf{D}$  by performing stochastic gradient descent.
- 4) Go to step 1 and iterate.

We call each cycle, i.e. each image patch has been trained once, as an *epoch*. Usually, several epochs are required to obtain a satisfactory result. When the number of image patches and the dictionary size is large, step 2 and step 3 are still very slow. We propose a novel algorithm to improve both of these parts, which is presented in the next section.

### III. STOCHASTIC COORDINATE CODING

In this section, we introduce our Stochastic Coordinate Coding (SCC) algorithm. It is known that solving the sparse coding problem usually is very time consuming especially when dealing with large-scale data sets and large size dictionaries [17]. The proposed algorithm aims to dramatically reduce the computational cost of the sparse coding while keeping comparable performance.

We detail our algorithm in the following. Initialize the dictionary via any initialization method and denote it as  $\mathbf{D}_1^1$ . Initialize the sparse code  $\mathbf{z}_i^0 = 0$  for  $i = 1, \dots, n$ . Here we use superscript to represent the number of epochs and we use subscript to represent the index of data points. Then starting from  $k = 1$  and  $i = 1$ , we do the following:

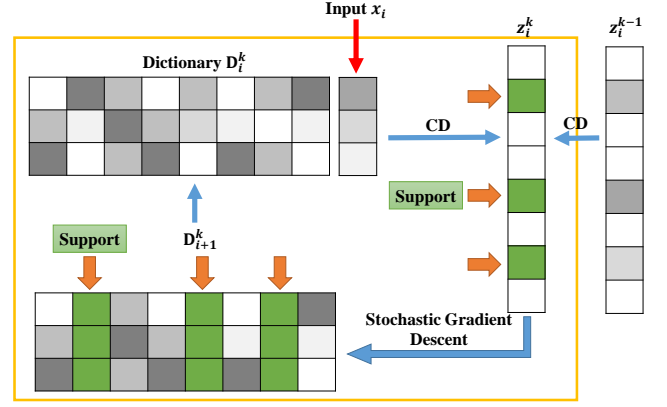


Fig. 2. Illustration of our algorithmic framework. With an image patch  $\mathbf{x}_i$ , we perform one step of coordinate descent to find the support sparse code. Next, we perform a few steps of coordinate descent on the support to obtain a new sparse code  $\mathbf{z}_i^k$ . Then we update the support of the dictionary by second order stochastic gradient descent to obtain a new dictionary  $\mathbf{D}_{i+1}^k$ .

- 1) Get an image patch  $\mathbf{x}_i$
- 2) Update  $\mathbf{z}_i^k$  via one or a few steps of coordinate descent:

$$\mathbf{z}_i^k = \text{CD}(\mathbf{D}_i^k, \mathbf{z}_i^{k-1}, \mathbf{x}_i). \quad (3)$$

Specifically, for  $j$  from 1 to  $m$ , we update the  $j$ -th coordinate  $z_{i,j}^{k-1}$  of  $\mathbf{z}_i^{k-1}$  cyclicly as follows:

$$b_j \leftarrow (\mathbf{d}_{i,j}^k)^T (\mathbf{x}_i - \mathbf{D}_i^k \mathbf{z}_i^{k-1}) + z_{i,j}^{k-1}, \\ z_{i,j}^{k-1} \leftarrow h_\lambda(b_j),$$

where  $h$  is the soft thresholding shrinkage function [10]. We call such one updating cycle as *one step* of coordinate descent. The updated sparse code is then denoted by  $\mathbf{z}_i^k$ . A detailed derivation of coordinate descent can be found in the appendix A.

- 3) Update the dictionary  $\mathbf{D}$  by using stochastic gradient descent:

$$\mathbf{D}_{i+1}^k = P_{B_m}(\mathbf{D}_i^k - \eta_i^k \nabla_{\mathbf{D}_i^k} f_i(\mathbf{D}_i^k, \mathbf{z}_i^k)), \quad (4)$$

where  $P$  denotes the projection operator. We set the learning rate as an approximation of the inverse of the Hessian matrix. The gradient of  $\mathbf{D}_i^k$  can be obtained as follows:

$$\nabla_{\mathbf{D}_i^k} f_i(\mathbf{D}_i^k, \mathbf{z}_i^k) = (\mathbf{D}_i^k \mathbf{z}_i^k - \mathbf{x}_i) (\mathbf{z}_i^k)^T.$$

- 4)  $i = i + 1$ . If  $i > n$ , then set  $\mathbf{D}_1^{k+1} = \mathbf{D}_{n+1}^k$ ,  $k = k + 1$  and  $i = 1$ .

We illustrate our algorithmic framework in Fig. (2). At each iteration, we get an image patch  $\mathbf{x}_i$ . Then we perform one or a few steps of coordinate descent to find the support of the sparse code. Next, we perform a few steps of coordinate descent on the support to obtain a new sparse code  $\mathbf{z}_i^k$ . Then we update the support of the dictionary by second order stochastic gradient descent.

It is known that the second step - updating the sparse code is the most time consuming part [2]. Coordinate descent

is known as one of the state of art methods for solving this lasso problem. Given an image patch  $\mathbf{x}_i$ , coordinate descent initialize  $\mathbf{z}_i^0 = 0$  and then update the sparse code many times via matrix-vector multiplication and thresholding. Empirically, the iteration may take tens hundreds steps to converge. However, we observed that after a few steps, the support of the coordinates, i.e., the locations of the nonzero entries in  $\mathbf{z}_i$ , is very accurate, usually less than ten steps. Note that the support of the sparse code is usually more important than the exact value of the sparse code. Moreover, since the original sparse coding is a non-convex problem and it involves an alternating updating, we do not need to run the coordinate descent to final convergence. Therefore, we propose to update the sparse code  $\mathbf{z}_i$  by using a few steps of coordinate descent. For the  $k$ -th epoch, we denote the updated sparse code as  $\mathbf{z}_i^k$ . It will be used as an initial sparse code for the  $k+1$ -th epoch.

After updating the sparse code, we know its support. One of our key insights is that when updating the dictionary, we can only need to focus on the support of the dictionary but not all columns of the dictionary. Let  $\mathbf{z}_{i,j}^k$  denote  $j$ -th entry of  $\mathbf{z}_i^k$  and let  $\mathbf{d}_{i,j}^k$  denote the  $j$ -th column of the dictionary  $\mathbf{D}_i^k$ . If  $\mathbf{z}_{i,j}^k = 0$ , then  $\nabla_{\mathbf{d}_{i,j}^k} f_i(\mathbf{D}_i^k, \mathbf{z}_i^k) = (\mathbf{D}_i^k \mathbf{z}_i^k - \mathbf{x}_i) \mathbf{z}_{i,j}^k = 0$ . Therefore,  $\mathbf{d}_{i,j}^k$  does not need to be updated. Assume  $\mathbf{z}_{i,j}^k$  is non-zero. Let  $\mathbf{d}_{i+1,j}^k$  denote the  $j$ -th column of the dictionary  $\mathbf{D}_{i+1}^k$ . Then we can update  $\mathbf{d}_{i+1,j}^k$  as follows:

$$\mathbf{d}_{i+1,j}^k \leftarrow \mathbf{d}_{i,j}^k - \eta_{i,j}^k \nabla_{\mathbf{d}_{i,j}^k} f_i(\mathbf{D}_i^k, \mathbf{z}_i^k) = \mathbf{d}_{i,j}^k - \eta_{i,j}^k \mathbf{z}_{i,j}^k (\mathbf{D}_i^k \mathbf{z}_i^k - \mathbf{x}_i), \quad (5)$$

Note that  $\mathbf{z}_i^k$  here is a sparse vector, therefore computing  $\mathbf{D}_i^k \mathbf{z}_i^k$  is very efficient. The computational cost will be significantly reduced when the support is very small. Note that for online dictionary learning, one usually has to update all columns of the dictionary. It is because that online dictionary learning uses the averaged gradient, which is usually not sparse. In other words, the support of the dictionary is itself. Therefore, one has to update all columns of the dictionary for each image patch. It is time consuming especially when the dictionary size is very large.

When the data sets are very large, the learning rate  $\eta_i^k$  will be very small after going through large number of image patches. In this case, the dictionary will not change very much and the efficiency of the training will decrease. In practice, turning the learning rate is very tricky and sensitive. In this paper, we use an adaptive learning rate. We aim to design a learning rate with the following two principals. The first one is that for different columns of the dictionary, we may use different learning rates. The second is that for the same column, the learning rate should decrease. Otherwise, the algorithm might not converge. To obtain the learning rate, we use the Hessian matrix of the objective function. It can be shown that the following matrix provides an approximation of the Hessian:  $\mathbf{H} = \sum_{k,i} \mathbf{z}_i^k (\mathbf{z}_i^k)^T$ , when  $k$  and  $i$  go to infinity. According to the second order stochastic gradient descent, we should use the inverse matrix of the Hessian as the learning rate. However, computing a matrix inversion problem is computationally expensive. In order to obtain the learning

rate, we simply use the diagonal element of the matrix  $\mathbf{H}$ . Note that if the columns of the dictionary have low correlation,  $\mathbf{H}$  is close to a diagonal matrix. Specifically, we first initialize  $\mathbf{H} = 0$ . Then update the matrix  $\mathbf{H}$  as follows:

$$\mathbf{H} \leftarrow \mathbf{H} + \mathbf{z}_i^k (\mathbf{z}_i^k)^T. \quad (6)$$

When updating the  $j$ -th column for the  $i$ th image patch  $\mathbf{x}_i$ , we replace  $\eta_{i,j}^k$  in Eq. (5) by  $1/h_{jj}$ , where  $h_{jj}$  is the  $j$ -th diagonal element of  $\mathbf{H}$ . In this way, we do not have to tune the learning rate parameter. It might be worth noting that we do not have to store the whole matrix of  $\mathbf{H}$  but only its diagonal elements. We summarize our algorithm in Algorithm 1.

---

**Algorithm 1** SCC (Stochastic Coordinate Coding)

---

**Require:** Data set  $\mathbf{X} = (\mathbf{x}_1 \cdots \mathbf{x}_n) \in \mathbb{R}^{p \times n}$

**Ensure:**  $\mathbf{D} \in \mathbb{R}^{p \times m}$  and  $\mathbf{Z} = (\mathbf{z}_1 \cdots \mathbf{z}_n) \in \mathbb{R}^{m \times n}$

**Initialize:**  $\mathbf{D}_1^1, \mathbf{H} = 0$  and  $\mathbf{z}_i^0 = 0$  for  $i = 1, \dots, n$ .

**for**  $k = 1$  **to**  $\kappa$  **do**

**for**  $i = 1$  **to**  $n$  **do**

        Get an image patch  $\mathbf{x}_i$

        Update  $\mathbf{z}_i^k$  via one or a few steps of coordinate descent:

$$\mathbf{z}_i^k \leftarrow \text{CD}(\mathbf{D}_i^k, \mathbf{z}_i^{k-1}, \mathbf{x}_i).$$

        Update the Hessian matrix and the learning rate:

$$\mathbf{H} \leftarrow \mathbf{H} + \mathbf{z}_i^k (\mathbf{z}_i^k)^T, \quad \eta_{i,j}^k = 1/h_{jj}.$$

        Update the support of the dictionary via SGD:

$$\mathbf{d}_{i+1,j}^k \leftarrow \mathbf{d}_{i,j}^k - \eta_{i,j}^k \mathbf{z}_{i,j}^k (\mathbf{D}_i^k \mathbf{z}_i^k - \mathbf{x}_i).$$

    If  $i = n$ , set  $\mathbf{D}_1^{k+1} = \mathbf{D}_{n+1}^k$ .

**end for**

**end for**

**Output**

$\mathbf{D} = \mathbf{D}_n^\kappa$  and  $\mathbf{z}_i = \mathbf{z}_i^\kappa$  for  $i = 1, \dots, n$ .

---

## IV. EXPERIMENTS

In this section, we empirically evaluate the efficiency and effectiveness of our proposed Stochastic Coordinate Coding (SCC) algorithm. A detailed description of data and experimental setting is given in Section IV-A. We study the influence of different settings of SCC in Section IV-B, including the influence of the number of coordinate descent steps and the learning rate. Finally, we compare SCC with the state-of-art sparse coding algorithm - Online dictionary Learning (OL, [21]) in terms of speed-up and accuracy in Section 5.3 and 5.4.

### A. Data Description and Experimental Setting

The Drosophila gene expression images used in our work are obtained from the FlyExpress database, which contains standardized images from the Berkeley Drosophila Genome Project (BDGP). The Drosophila embryogenesis is partitioned into 6 stage ranges (1-3, 4-6, 7-8, 9-10, 11-12, 13-17) in BDGP. We focus on the later 5 stage ranges as there are few keywords appeared in the first stage range.

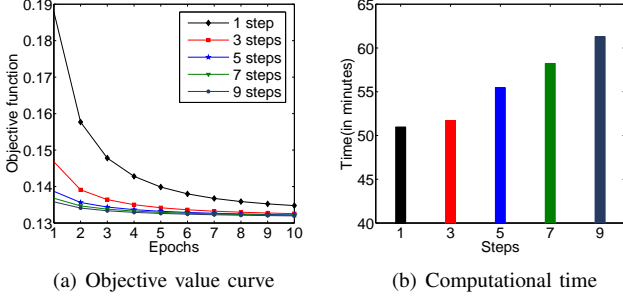


Fig. 3. A comparison of different coordinate descent steps. (a) shows the objective value curves when varying the number of coordinate descent steps. The horizontal axis represents the number of epochs. (b) shows the computational time (in minutes) of running 10 epochs. It can be seen from the figure that using a great number of coordinate descent steps can achieve lower objective value. However, the overall computational time would increase.

The *Drosophila* embryos are 3D objects [36], and the FlyExpress database contains 2D images that are taken from different views (lateral, dorsal, and lateral-dorsal) [20]. As majority of images in the database are in lateral view [15], we focus on the lateral-view images in our study. For each image, we first use a  $16 \times 16$  window to obtain a collection of small image patches. Then we extract a 128-dimensional Scale-Invariant Feature Transform (SIFT, [19]) feature from each image patch. Each SIFT feature is further normalized to be zero mean and unit  $l_2$  norm. The patches with small standard deviations were discarded. After preprocessing the data, we have 555009, 259882, 286349, 989653, 1006012 image patches for different stage ranges (4-6, 7-8, 9-10, 11-12, 13-17) respectively.

For each state range, we first initialize the dictionary via selecting random patches [9], which has been shown to be a very efficient and effective initialization method in practice. Then we learn the sparse codes by different sparse coding methods using the same initial dictionary. All 5 stage ranges will be trained for 10 epochs using a batch size of 1. After learning the sparse codes, we apply max pooling [27] to generate the features for annotation. Finally, we employ the one-against-rest support vector machines (SVM, [7]) to annotate the gene expression pattern images.

The regularization parameter  $\lambda$  is set to  $0.10 = 1.2/\sqrt{p}$  in all of our experiments. The  $1/\sqrt{p}$  term comes from a classical normalization factor [4], and the constant 1.2 has shown to produce about 10 non-zero coefficients for *Drosophila* data sets. We have implemented the proposed algorithm in C++ and all the experiments have been run on a single-CPU, eight-core 3.4Ghz machine.

## B. Model Selection

In this section, we study the influence of different algorithm settings, including the number of coordinate descent steps and learning rates.

1) *The Number of Coordinate Descent Steps*: First, experiments were carried out to study the influence of the number of coordinate descent steps. We use the state range 2 in our

experiments. It has 555009 image patches and each image patch is of 128 dimensions. The dictionary size is  $1000 \times 128$ . We tested 1, 3, 5, 7, 9 steps of coordinate descent. The results are evaluated by the objective function value and the running time, as shown in Fig. 3.

It can be seen from Fig. (3) that using a great number of coordinate descent steps can achieve lower objective function value, however the computational time would increase. Therefore we should choose a suitable number of coordinate descent steps. In practice, we choose 3 steps of coordinate descent, which performs quit well in all experiments. Also, we can see from Fig. (3)(a) that SCC converges under coordinate descent steps.

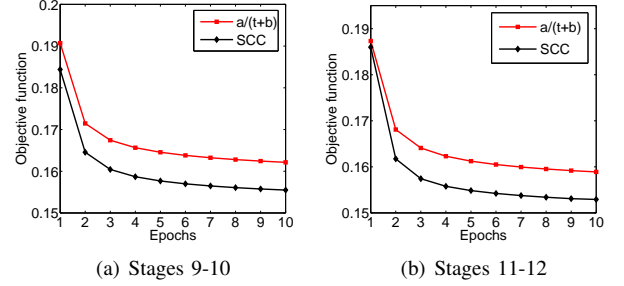


Fig. 4. A comparison of different learning rates on two different stage ranges. From the figure we can see that the adaptive learning rate consistently performs better than the natural learning rates. Moreover, the adaptive learning rate converges faster than the natural learning rate.

2) *Learning Rates*: We also studied the influence of the learning rate. We compared our adaptive learning rate with the natural learning rate on two different stage ranges. The natural learning rate is given as  $\eta_t = \frac{a}{t+b}$ , where  $a$  and  $b$  are predefined parameters. We exhaustively search for the best parameter settings among  $a \in [10^{-3}, 10^3]$  and  $b \in [10^{-3}, 10^3]$  as determined by their lowest objective function value. We present the optimal result of the natural learning rate. It can be seen from the Fig. 4 that the adaptive learning rate consistently performs better than the natural learning rate.

## C. Computational Time Comparison

We first show the computational time of updating the dictionary and updating the sparse code. Table II shows the computational time of these two steps on three different dictionary sizes, i.e.,  $500 \times 128$ ,  $1000 \times 128$  and  $2000 \times 128$ . It can be seen from the table that SCC significantly reduces the computational time. Note that when the size of the dictionary increases, the computational time of OL increases rapidly. However, for SCC the computational time increases much slower compared to OL, especially the computational time of updating the dictionary. Therefore, SCC has a better scalability when dealing with large size dictionaries.

A computational time as well as an objective function value comparison is given in Table I. It can be seen from the table that SCC archives a very low objective function value, which is comparable with OL. Meanwhile, the computational time

TABLE I  
A COMPARISON OF SCC AND OL ON OBJECTIVE FUNCTION VALUES AND COMPUTATIONAL TIME.

128 × 500	Stages	4-6	7-8	9-10	11-12	13-17
Objective function value	OL	0.1374	0.1495	0.1470	0.1465	0.1489
	SCC	0.1384	0.1503	0.1478	0.1474	0.1498
Running time (in hours)	OL	15.71	7.81	8.60	30.64	31.14
	SCC	0.1439	0.0680	0.0748	0.2573	0.2616
128 × 1000	Stages	4-6	7-8	9-10	11-12	13-17
Objective function value	OL	0.1317	0.1420	0.1398	0.1399	0.1424
	SCC	0.1325	0.1429	0.1407	0.1408	0.1433
Running time (in hours)	OL	59.61	28.31	30.77	107.80	111.01
	SCC	0.1889	0.0900	0.0975	0.3399	0.3469
128 × 2000	Stages	4-6	7-8	9-10	11-12	13-17
Objective function value	OL	0.1266	0.1358	0.1338	0.1342	0.1365
	SCC	0.1278	0.1366	0.1349	0.1353	0.1378
Running time (in hours)	OL	219.21	102.64	113.09	390.88	397.34
	SCC	0.3666	0.1648	0.1902	0.6397	0.6401

of SCC is much less than OL. Note that when the dictionary size increases, the objective function value decreases.

In this work we focus on the the single batch size setting, that is, we process one image patch in each iteration. We also compare our proposed SCC (with a batch size of 1) with mini-batch OL (with a batch size of 512). Our empirical results show that the mini-batch OL is about 3-4 times faster than SCC. Note that the mini-batch algorithms are usually faster than incremental algorithms. In addition, the implementation of mini-batch OL (SPAM<sup>1</sup>) is well optimized, making a direct time comparison difficult. We plan to develop the mini-batch extension of SCC and further optimize the code to improve its efficiency.

TABLE II  
A COMPUTATIONAL TIME COMPARISON (IN HOURS) OF SCC AND OL FOR DIFFERENT DICTIONARY SIZES.

Dictionary Sizes	OL			SCC		
	500	1000	2000	500	1000	2000
Update <b>Z</b>	2.58	8.83	38.75	0.22	0.39	0.71
Update <b>D</b>	5.23	19.48	63.89	0.03	0.03	0.04
Total	7.81	28.31	102.64	0.25	0.42	0.75

#### D. Annotation Performance Comparison

In this experiment, we compare the results of *Drosophila* gene image annotation by using learned features from SCC and OL. We tested all 5 stage ranges with different dictionary sizes, i.e.,  $500 \times 128$ ,  $1000 \times 128$  and  $2000 \times 128$ . We choose four measurements: accuracy, AUC, sensitivity and specificity to evaluate the performance of different approaches. Comparison results for all 5 stage ranges by a weighted average of the top 10 terms are shown in Fig. 5.

It can be seen from the figure that when the dictionary size is  $500 \times 128$ , OL performs slightly better SCC. When the dictionary size is  $1000 \times 128$  or  $2000 \times 128$ , SCC and OL achieve comparable results. However, SCC is significantly

faster than OL in this case. It might be worth noting that when the dictionary size increases, SCC and OL both improve their performance.

#### V. CONCLUSIONS

In this paper, we propose a new algorithm called Stochastic Coordinate Coding (SCC) to solve the sparse coding problem. In SCC, we perform a few steps of coordinate descent to update the sparse codes and use second order stochastic gradient descent to update the dictionary. The computational cost is further reduced by only updating the support of the sparse codes and the dictionary. Extensive experiments on *Drosophila* gene expression data sets have demonstrated the efficiency of the proposed algorithm. Compared to the state-of-art sparse coding algorithms, the proposed algorithm achieves one or two orders of magnitude speed-up (see Table I) when varying the dictionary size. The idea of combining coordinate descent and stochastic gradient descent can be applied to other problem settings. For example, we can extend the algorithm to solve the supervised sparse coding problem [22]. In addition, we plan to extend the algorithm from the single task learning setting to the multiple task learning setting. We also plan to develop the mini-batch implementation of SCC.

#### REFERENCES

- [1] S. Balakrishnan and D. Madigan. Algorithms for sparse linear classifiers in the massive data setting. *Journal of Machine Learning Research*, 9:313–337, 2008.
- [2] K. Balasubramanian, K. Yu, and G. Lebanon. Smooth sparse coding via marginal regression for learning sparse representations. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 289–297, 2013.
- [3] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Img. Sci.*, 2(1):183–202, 2009.
- [4] P. J. Bickel, Y. Ritov, and A. B. Tsybakov. Simultaneous analysis of lasso and dantzig selector. *The Annals of Statistics*, pages 1705–1732, 2009.
- [5] L. Bottou. Online learning and stochastic approximation. *Online Learning and Neural Networks*, Cambridge University Press, Cambridge, UK, 1998.
- [6] L. Bottou and O. Bousquet. 13 the tradeoffs of large-scale learning. *Optimization for Machine Learning*, page 351, 2011.

<sup>1</sup>We use the code from the authors downloaded from the SPAM package: <http://spams-devel.gforge.inria.fr/>.



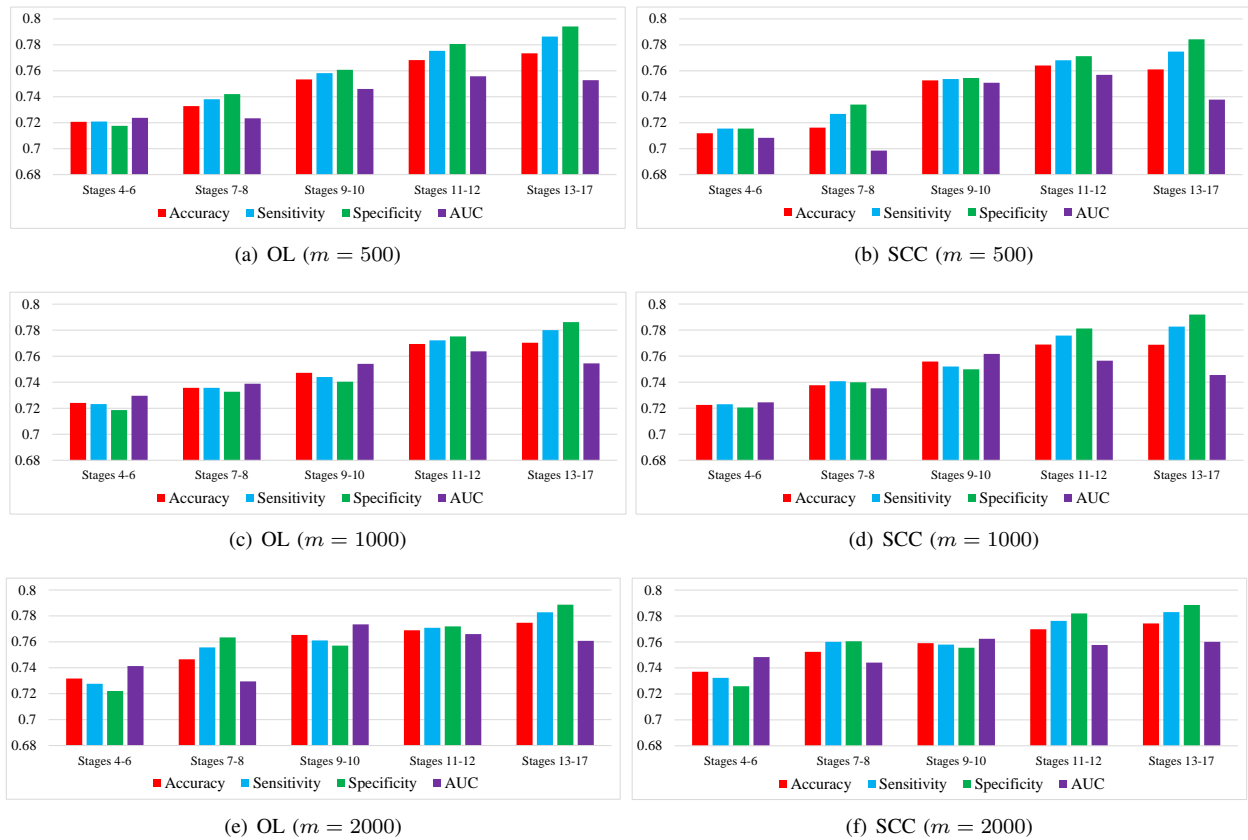


Fig. 5. A comparison of SCC and OL on image annotation. (a)-(b), (c)-(d) and (e)-(f) show the image annotation results by using different dictionary sizes  $500 \times 128$ ,  $1000 \times 128$  and  $2000 \times 128$ . It can be seen from the figure that when the dictionary size is  $500 \times 128$ , OL performs slightly better SCC. When the dictionary size is  $1000 \times 128$  or  $2000 \times 128$ , SCC and OL achieve comparable results.

- [7] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.
- [8] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM journal on scientific computing*, 20(1):33–61, 1998.
- [9] A. Coates and A. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 921–928, New York, NY, USA, June 2011. ACM.
- [10] P. L. Combettes and V. R. Wajs. Signal Recovery by Proximal Forward-Backward Splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200, 2005.
- [11] D. L. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via  $\ell_1$  minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202, 2003.
- [12] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.
- [13] G. Grumblin, V. Strelets, and The FlyBase Consortium. FlyBase: anatomical data, images and queries. *Nucleic Acids Research*, 34:D484–488, 2006.
- [14] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, pages 2146–2153, 2009.
- [15] S. Ji, L. Yuan, Y.-X. Li, Z.-H. Zhou, S. Kumar, and J. Ye. Drosophila gene expression pattern annotation using sparse features and term-term interactions. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 407–416. ACM, 2009.
- [16] S. Ji, L. Yuan, Y.-X. Li, Z.-H. Zhou, S. Kumar, and J. Ye. Drosophila gene expression pattern annotation using sparse features and term-term interactions. In *Proceedings of the Fifteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 407–416, 2009.
- [17] H. Lee, A. Battle, R. Raina, and A. Ng. Efficient sparse coding algorithms. In *Advances in neural information processing systems*, pages 801–808, 2006.
- [18] M. Levine and E. H. Davidson. Gene regulatory networks for development. *Proceedings of the National Academy of Sciences of the United States of America*, 102(14):4936–4942, 2005.
- [19] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.
- [20] D. L. Mace, N. Varnado, W. Zhang, E. Frise, and U. Ohler. Extraction and comparison of gene expression patterns from 2d rna in situ hybridization images. *Bioinformatics*, 26(6):761–769, 2010.
- [21] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 689–696. ACM, 2009.
- [22] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised dictionary learning. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1033–1040. 2009.
- [23] K. A. Matthews, T. C. Kaufman, and W. M. Gelbart. Research resources for drosophila: the expanding universe. *Nat Rev Genet*, 6(3):179–193, Mar. 2005.
- [24] B. A. Olshausen et al. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.
- [25] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.
- [26] B. A. Olshausen and D. J. Field. Sparse coding of sensory inputs. *Current opinion in neurobiology*, 14(4):481–487, 2004.
- [27] D. Scherer, A. Müller, and S. Behnke. Evaluation of pooling operations

- in convolutional architectures for object recognition. In *Artificial Neural Networks-ICANN 2010*, pages 92–101. Springer, 2010.
- [28] S. W. Sean Carroll, Jennifer Grenier. *From DNA to diversity : molecular genetics and the evolution of animal design*, volume 2nd Edition. Wiley-Blackwell, 2005.
- [29] E. Smith and M. Lewicki. Efficient auditory coding. *Nature*, 439:978–982, February 2006.
- [30] Q. Sun, S. Muckatira, L. Yuan, S. Ji, S. Newfeld, S. Kumar, and J. Ye. Image-level and group-level models for drosophila gene expression pattern annotation. *BMC bioinformatics*, 14(1):350, 2013.
- [31] A. Szlam, K. Gregor, and Y. LeCun. Fast approximations to structured sparse coding and applications to object classification. In *ECCV (5)*, pages 200–213, 2012.
- [32] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*, 58:267–288, 1996.
- [33] R. Tibshirani, J. Bien, J. Friedman, T. Hastie, N. Simon, J. Taylor, and R. J. Tibshirani. Strong rules for discarding predictors in lasso-type problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(2):245–266, 2012.
- [34] P. Tomancak, A. Beaton, R. Weiszmann, E. Kwan, S. Shu, S. E. Lewis, S. Richards, M. Ashburner, V. Hartenstein, S. E. Celniker, and G. M. Rubin. Systematic determination of patterns of gene expression during *Drosophila* embryogenesis. *Genome Biology*, 3(12), 2002.
- [35] P. Tomancak, B. Berman, A. Beaton, R. Weiszmann, E. Kwan, V. Hartenstein, S. Celniker, and G. Rubin. Global analysis of patterns of gene expression during *Drosophila* embryogenesis. *Genome Biology*, 8(7):R145, 2007.
- [36] G. H. Weber, O. Rubel, M.-Y. Huang, A. H. DePace, C. C. Fowlkes, S. V. Keranen, C. L. Luengo Hendriks, H. Hagen, D. W. Knowles, J. Malik, et al. Visual exploration of three-dimensional gene expression using physical views and linked abstract views. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 6(2):296–309, 2009.
- [37] T. T. Wu and K. Lange. Coordinate Descent Algorithms for Lasso Penalized Regression. *The Annals of Applied Statistics*, 2(1):224–244, 2008.
- [38] L. Yuan, A. Woodard, S. Ji, Y. Jiang, Z.-H. Zhou, S. Kumar, and J. Ye. Learning sparse representations for fruit-fly gene expression pattern image annotation and retrieval. *BMC bioinformatics*, 13(1):107, 2012.
- [39] J. Zhou and H. Peng. Automatic recognition and annotation of gene expression patterns of fly embryos. *Bioinformatics*, 23(5):589–596, 2007.

## APPENDIX

Given a data point  $\mathbf{x} = (x_1, \dots, x_p)^T \in \mathbb{R}^p$  and a dictionary  $\mathbf{D} \in \mathbb{R}^{m \times p}$ , the lasso problem is given as follows:

$$\min_{\mathbf{z}} f(\mathbf{z}) = \frac{1}{2} \|\mathbf{D}\mathbf{z} - \mathbf{x}\|_2^2 + \lambda \|\mathbf{z}\|_1, \quad (7)$$

where  $\mathbf{z} = (z_1, \dots, z_m)^T \in \mathbb{R}^m$ .

If we freeze all components of  $\mathbf{z}$  except the  $j$ -th column  $z_j$  in Eq. (7). Let  $\mathbf{d}_j$  denote the  $j$ -th column of  $\mathbf{D}$ ,  $d_{ij}$  the element of  $\mathbf{D}$  in the  $i$ -th row and  $j$ th column. We have

$$\begin{aligned} & \operatorname{argmin}_{z_j} \frac{1}{2} \sum_{i=1}^p \left( \sum_{j=1}^m d_{ij} z_j - x_i \right)^2 + \lambda \sum_{j=1}^m |z_j| \\ &= \operatorname{argmin}_{z_j} \frac{1}{2} (z_j^2 - 2b_j z_j + \|\mathbf{x}\|_2^2) + \lambda |z_j| \\ &= \operatorname{argmin}_{z_j} \frac{1}{2} (z_j - b_j)^2 + \lambda |z_j|, \end{aligned}$$

where  $b_j = \sum_{i=1}^p d_{ij} (x_i - \sum_{k \neq j} d_{ik} z_k)$  and we have used the condition that each column of  $\mathbf{D}$  is unit norm. Then  $z_j$  has an explicit optimal solution:  $z_j = h_\lambda(b_j)$ , where  $b_j = \sum_{i=1}^p d_{ij} (x_i - \sum_{k \neq j} d_{ik} z_k)$  and  $h$  is a soft thresholding

shrinkage function or called the proximal operator of the  $l_1$  norm [10]. It is defined as

$$h_\lambda(v) = \begin{cases} v + \lambda, & v < -\lambda \\ 0, & -\lambda \leq v \leq \lambda \\ v - \lambda, & \lambda < v \end{cases}$$

Note that  $b_j = \sum_{i=1}^p d_{ij} (x_i - \sum_{k \neq j} d_{ik} z_k) = \mathbf{d}_j^T \mathbf{x} - \mathbf{d}_j^T \mathbf{D}\mathbf{z} + (\mathbf{d}_j^T \mathbf{d}_j) z_j = \mathbf{d}_j^T (\mathbf{x} - \mathbf{D}\mathbf{z}) + z_j$ . Therefore, the computational cost of updating the  $j$ -th coordinate  $z_j$  depends on computing the vector  $\mathbf{r} = \mathbf{x} - \mathbf{D}\mathbf{z}$  and the inner product  $\mathbf{d}_j^T \mathbf{r}$ .